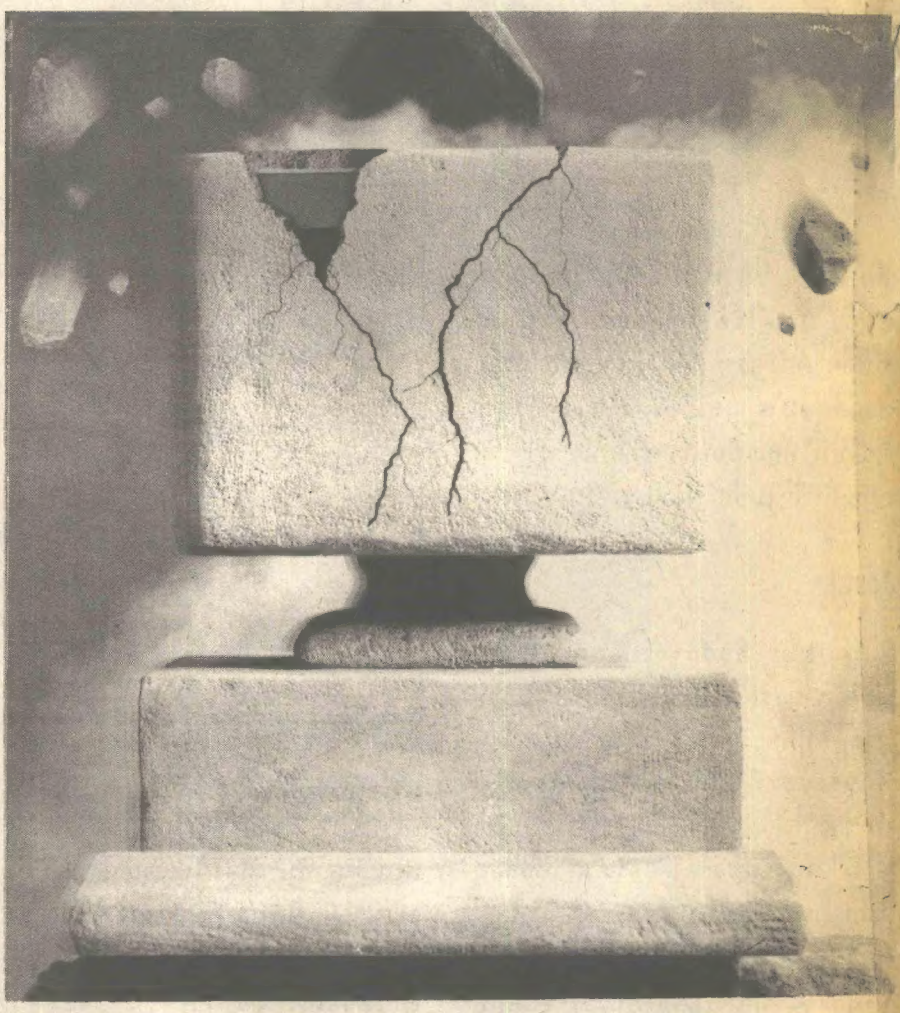
อธิการ มูลสาร

SOFT MAIL

ดอสยูสเซอร์ : เขียนเมนูแถบแสง (light-bar-menu) ด้วยแบตช์ไฟล์



ภาษาโปรแกรมไม่ใช่สิ่งสำคัญ มันอาจจะทำให้เราเข้าใจการทำงานของโปรแกรมง่ายขึ้นแต่สิ่งที่สำคัญคือ logic ในการแก้ปัญหาต่างหาก

สมมติแผ่นดอสของคุณไม่มีโปรแกรมภาษาเบสิก คุณสามารถเขียนโปรแกรมใช้งานด้วยโปรแกรม debug ที่ให้มากับดอสคุณสามารถเขียนภาษาแอสเซมบลีสั้น ๆ แต่ทำงานมีประสิทธิภาพได้ พร้อมกับการใช้งานแบตช์ไฟล์ที่เขียนขึ้นเอง

ข้อตกลงของเรา

ต่อนี้ไป ข้อความที่อยู่ในเครื่องหมายวงเล็บปีกกา ({ }) เราจะถือเสมือนว่าเป็นอักขระพิเศษ ซึ่งเป็นอักขระเพียงตัวเดียว เช่น {esc} คืออักขระที่ 27 ตามตารางอักขระแอสกี ไม่ได้หมายความว่าให้ป้อน {, E, s, c และ } รวม 5 ตัวอักษร ข้อความที่อยู่ในเครื่องหมาย < > เราจะถือเสมือนการกดปุ่มที่มีชื่อในเครื่องหมายดังกล่าวเพียงปุ่มเดียวหรือกดควบกันเพียงครั้งเดียว เช่น <Enter> จะหมายถึงการกดปุ่ม Enter เพียงครั้งเดียว หรือ <Ctrl-Q> จะหมายถึงการกดปุ่ม <Ctrl> ค้างไว้ตามด้วยปุ่ม <Q> แล้วจึงปล่อยปุ่มทั้งสองพร้อมกัน เหล่านี้เป็นต้น

REPLY.COM

เราจะไม่เขียนโปรแกรมที่ยาวเหยียด เพราะดีบักไม่เหมาะกับอย่างนั้น สิ่งที่เราต้องการ (แต่ไม่มีให้) คือ คำสั่งหรือโปรแกรมอ

รูปที่ 1

```
A 100
MOV  AH, 8
INT  21
CMP  AL, 0
JNZ  10A
INT  21
<Enter>
R CX
E
N REPLY.COM
W
Q
```

ตารางที่ 1

อีดิเตอร์	ปุ่มนำ	ตามด้วย	ผลบนหน้าจอ*
SideKick, TURBO family**	<Ctrl-P>	<Esc>	[
Norton family	<Ctrl-Q>	<Esc>	<-
Quick family	<Ctrl-P>	<Esc>	<-
EDLIN	<Ctrl-[>	<[>	â[[

* ผลบนหน้าจอจะเห็นได้ชัดเนื่องจากจะแสดงด้วยสีที่ต่างไปจากสีปกติ หรือมีแอตทริบิวต์เป็นตัวอักษรจาง ๆ

** family หมายถึงอีดิเตอร์ของคอมไพเลอร์ตระกูลเดียวกัน หรือมีลักษณะเหมือนกัน

```
@echo off
: loop
reply
if errorlevel 59 if not errorlevel 60 echo F1-Key
if errorlevel 60 if not errorlevel 61 echo F2-Key
if errorlevel 61 if not errorlevel 62 echo F3-Key
if errorlevel 62 if not errorlevel 63 echo F4-Key
if errorlevel 63 if not errorlevel 64 echo F5-Key
if errorlevel 64 if not errorlevel 65 echo F6-Key
if errorlevel 65 if not errorlevel 66 echo F7-Key
if errorlevel 66 if not errorlevel 67 echo F8-Key
if errorlevel 67 if not errorlevel 68 echo F9-Key
if errorlevel 68 if not errorlevel 69 echo F10-Key
goto loop
```

รูปที่ 2

สักอย่าง ที่จะสามารถส่งค่ากลับมาที่ดอส เมื่อผู้ใช้กดคีย์บอร์ด ซึ่งเป็นค่าที่บอกได้ว่า เรากดคีย์อะไร เราสามารถตรวจสอบได้โดยใช้คำสั่ง if errorlevel ต่อไปนี้ให้คุณป้อนข้อความดังรูปที่ 1 นี้ แล้วเก็บไว้ในชื่อไฟล์ว่า REPLY.DBG

{esc}

{esc} คืออักขระแอสกีตัวที่ 27 ซึ่งไม่สามารถป้อนลงในแฟ้มข้อมูลด้วยการกดปุ่มธรรมดาได้ เราสามารถป้อนอักขระแอสกีได้ทั้ง 255 ตัว (ยกเว้นอักขระที่ 0 หรือ null) บางส่วนเราป้อนได้ทางคีย์บอร์ดโดยตรง แต่บางส่วนไม่สามารถป้อนทางคีย์บอร์ดได้ (non-keyboard character) เราก็สามารถใช้ปุ่ม <Alt> (Alternate) ร่วมกับกลุ่มตัวเลขทางขวามือ (numeric key-pad) เพื่อป้อนอักขระดังกล่าวได้ครบทุกตัว

อักขระบางตัว เช่น อักขระตัวที่ (Enter) หรืออักขระที่ 27 (Escape) จะตอบสนองทันทีที่อักขระดังกล่าวป้อนลงไป ไม่ว่าเราจะกด <Esc>, <Alt-27> หรือ <Ctrl-[> จะได้ผลแบบเดียวกัน ดอสให้เราสามารถป้อนรหัส {esc} ได้โดยใช้โปรแกรม EDLIN ของดอส วิธีป้อนคือกด <Cltrl-V> ตามด้วยเครื่องหมาย "[" จะถือว่าเป็นรหัส {esc} ได้ ในโปรแกรม editor ตัวอื่น ๆ ดูรายละเอียดจากตารางที่ 1

เริ่มเขียนเมนู

เราจะเริ่มเขียนเมนูแถบแสง แต่ก่อนอื่นเราลองมาทดสอบโปรแกรม REPLY.COM ที่ได้ ลองป้อนแบตช์ไฟล์ตามรูปที่ 2

REPLY.COM จะหยุดให้ผู้ใช้กดปุ่มใด ๆ และจะรีเทอร์นค่าต่าง ๆ กลับมา เราสามารถตรวจสอบค่าดังกล่าวได้ด้วยคำสั่ง if errorlevel โดยค่าของ F1-F10 จะอยู่ระหว่าง 59-68 ตามลำดับ การใช้คำสั่ง if errorlevel จะไล่จากสูงไปต่ำ ทั้งนี้เนื่องจาก errorlevel ที่สูงกว่าหรือเท่ากับที่ตรวจสอบถือว่าเป็นจริงทั้งหมด โปรแกรมนี้จะไม่หยุดทำงานจนกว่าผู้จะกด <Ctrl-C> หรือ

<Ctrl-Break>

สมมติว่าเราตรวจสอบคีย์ด้วยแบตช์ไฟล์ดังรูปที่ 3

ในกรณีที่เรากด <F2> แบตช์ไฟล์จะแสดงข้อความ "F2-Key" ออกมา แต่ถ้าเรากด <F1> แบตช์ไฟล์จะแสดงทั้งข้อความ "F1" และ "F2" ทั้งนี้เพราะ if errorlevel จะตรวจสอบค่าคีย์ที่เท่ากับหรือสูงกว่า วิธีแก้ไขคือให้กลับลำดับการตรวจสอบจากมากไปน้อย แต่วิธีที่สะดวกและดีที่สุดคือใช้คำสั่ง if errorlevel...if not errorlevel... อย่างต่อเนื่องโดยจากตัวอย่าง ถ้าเราต้องการตรวจสอบปุ่ม <F1> ให้ใช้คำสั่งว่า if errorlevel 59 if not errorlevel 60...

MENU.BAT

ลองพิมพ์แบตช์ไฟล์ตามรูปที่ 4 สำหรับรหัส {esc} วิธีป้อนได้กล่าวไว้ตอนต้นแล้ว จะไม่นำมากล่าวถึงอีก แต่ขอย้ำความเข้าใจว่า ไม่ใช่การป้อน {, e, s, c และ } แต่เป็นอักขระเพียงตัวเดียว

ข้อความที่มีคำว่า times ตามท้ายหมายถึงป้อนอักขระดังกล่าวซ้ำ ๆ กัน n ครั้ง เช่น <Alt-219>7times หมายถึงป้อน <Alt-219> 7 ครั้ง

หมายเลขบรรทัดมีไว้เพื่ออ้างอิง

รูปที่ 3

```
@echo off
: loop
reply
if errorlevel 59 echo F1-Key
if errorlevel 60 echo F2-Key
goto loop
```

รูปที่ 4

```
 1: @echo off
 2: set msg1=dBASE III Plus version 1.0
 3: set msg2=Lotus 1-2-3 release 2.01
 4: set msg3=Rajvithi Word PC version 2.0
 5: set msg=%msg1%
 6: set row=12
 7:
 8: :ShowMenu
 9: echo {esc}[2J
10: echo {esc}[07;14H<Alt-219>54times
11: echo {esc}[08;14H<Alt-219>54times
12: echo {esc}[09;14H<Alt-219>7times
    <Alt-219>7times
13: echo {esc}[10;14H<Alt-219>7times    <<<<<  Main Menu  >>>>>
    {esc}[7m<Alt-176>2times{esc}[m<Alt-219>7times
14: echo {esc}[11;14H<Alt-219>7times
    {esc}[7m<Alt-176>2times{esc}[m<Alt-219>7times
15: echo {esc}[12;14H<Alt-219>7times  %msg1%
    {esc}[7m<Alt-176>2times{esc}[m<Alt-219>7times
16: echo {esc}[13;14H<Alt-219>7times  %msg2%
    {esc}[7m<Alt-176>2times{esc}[m<Alt-219>7times
17: echo {esc}[14;14H<Alt-219>7times  %msg3%
    {esc}[7m<Alt-176>2times{esc}[m<Alt-219>7times
18: echo {esc}[15;14H<Alt-219>7times
    {esc}[7m<Alt-176>2times{esc}[m<Alt-219>7times
19: echo {esc}[16;14H<Alt-219>9times{esc}[7m<Alt-176>38times{esc}[m
    <Alt-219>7times
20: echo {esc}[17;14H<Alt-219>54times
21: echo {esc}[18;14H<Alt-219>54times
22: echo {esc}[20;14H Press {esc}[7mEsc{esc}[m to Quit or {esc}[7m
    Enter{esc}[m to select.
23:
24: :HiLight
25: echo {esc}[%row%; 23H{esc}[7m %msg%{esc}[m
26:
27: :Reply
28: echo {esc}[1;1H
29: reply
30: if errorlevel 80 if not errorlevel 81 goto ArrowDn
31: if errorlevel 72 if not errorlevel 73 goto ArrowUp
32: if errorlevel 27 if not errorlevel 28 goto End
33: if errorlevel 13 if not errorlevel 14 goto Enter
```

รูปที่ 4 (ต่อ)

```
34: echo <Alt-7>
35: goto Reply
36:
37: :ArrowDn
38: echo [esc][%row%;23H %msg%
39: if %row% = = 14 goto WrapUp
40: if %row% = = 13 set msg = %msg3%
41: if %row% = = 13 set row = 14
42: if %row% = = 12 set msg = %msg2%
43: if %row% = = 12 set row = 13
44: goto HiLight
45:
46: :WrapUp
47: set msg = %msg1%
48: set row = 12
49: goto HiLight
50:
51: :ArrowUp
52: echo [esc][%row%;23H %msg%
53: if %row% = = 12 goto WrapDn
54: if %row% = = 13 set msg = %msg1%
55: if %row% = = 13 set row = 12
56: if %row% = = 14 set msg = %msg2%
57: if %row% = = 14 set row = 13
58: goto HiLight
59:
60: :WrapDn
61: set msg = %msg3%
62: set row = 14
63: goto HiLight
64:
65: :Enter
66: if %row% = = 13 call dbase.bat
67: if %row% = = 14 call lotus.bat
68: if %row% = = 15 call rw.bat
69: goto ShowMenu
70:
71: :End
72: set msg =
73: set row =
74: set msg1 =
75: set msg2 =
76: set msg3 =
77: echo [esc][2J
78: echo * End - See you again!
```

■

และอธิบายเท่านั้น กรุณาอย่าป้อนเข้าไปด้วย

ANSI.SYS

ANSI.SYS เป็นดีไวซ์ไดรเวอร์ (โปรแกรม) ควบคุมการแสดงผล เราจะติดตั้ง ANSL.SYS ได้โดยใช้คำสั่ง

```
DEVICE = ANSI.SYS
```

ในไฟล์ CONFIG.SYS ตรวจดูว่ามีคำสั่งข้างต้นบรรจุอยู่ด้วยหรือไม่ ถ้าไม่ให้เพิ่มเข้าไปและเซฟไฟล์ CONFIG.SYS ใหม่นี้ บูตเครื่องใหม่ ดอสจะโหลด ANSI.SYS ลงไปไว้ในหน่วยความจำ

การกำหนดตำแหน่งแสดงผล

ANSI.SYS เอื้ออำนวยให้เราสามารถกำหนดตำแหน่งแสดงผลได้ด้วยคำสั่ง echo หรือคำสั่ง type โดยใช้รหัส [esc] เป็นอักษรนำ ตามด้วย "[" และปิดท้ายด้วย H (ตัวพิมพ์ใหญ่เท่านั้น) มีรูปแบบดังนี้

```
[esc][<row>;<col>H ข้อความ...
```

สมมติว่าเราต้องการแสดงข้อความ Hello! ที่บรรทัดแรกคอลัมน์ที่ 70 ให้ใช้คำสั่ง echo [esc] [1;70HHello! หรือเก็บข้อความตั้งแต่รหัส [esc] จนถึง ! ไว้ในไฟล์แล้ว type หรือ copy ไฟล์ดังกล่าวออกมาที่หน้าจอ ANSI.SYS จะแสดงข้อความ Hello! ได้อย่างถูกตำแหน่ง

ค่าของ <row> อยู่ระหว่าง 1-25 อย่างไรก็ตาม คุณไม่สามารถแสดงข้อความที่บรรทัดที่ 25 ได้เลย เนื่องจากคำสั่ง echo จะเลื่อนเคอร์เซอร์

มาที่ต้นบรรทัดใหม่หลังแสดงข้อความแล้วเสมอ ส่วนค่าของ <col> จะอยู่ระหว่าง 1-80 ขอแนะนำให้คุณระวังเรื่องการแสดงผลที่คอลัมน์ 80 ด้วย ปัญหาเรื่องเคอร์เซอร์เช่นเดียวกับปัญหาของบรรทัดที่ 25 บนหน้าจอ

แอตทริบิวต์

เราสามารถกำหนดแอตทริบิวต์ของการแสดงผลได้โดยชุดคำสั่ง

```
[esc][<attr>m
```

โดยค่าของ <attr> สำหรับจอโมโนโครม มีดังนี้

```
0 ปกติ
1 เข้ม
4 ขีดเส้นใต้
5 กระพริบ
7 กลับขาวดำ (reverse)
8 ไม่แสดงผล (invisible)
```

สำหรับค่าแอตทริบิวต์ของจอสีดูได้จากตารางที่ 2

เราจะลองแสดงข้อความด้วยแอตทริบิวต์ต่าง ๆ ข้างต้น แบตช์ไฟล์ตัวอย่างต่อไปนี้ จะใช้สำหรับจอโมโนโครม สำหรับจอสี เพียงแต่เปลี่ยนค่า <attr> ไปเป็นแอตทริบิวต์สำหรับจอสีเท่านั้น

คำสั่งกำหนดแอตทริบิวต์ ถ้าไม่กำหนดหมายเลขแอตทริบิวต์ ANSI.SYS จะกำหนดให้เป็นแอตทริบิวต์ปกติของจอ

ข้อความ INVISIBLE จะไม่แสดงออกทางจอภาพถ้าแบตช์ไฟล์ของเราทำงานถูกต้อง ตรวจสอบผลการทำงานบนจอภาพ แล้วลองผสมผสานแอตทริบิวต์ต่าง ๆ โดยใช้เครื่องหมาย ";" คั่นหมายเลข <attr> แต่ละตัว ดังโปรแกรมในรูปที่ 5

ตัวอักษร	แบคกราวนด์	สี
30	40	ดำ
31	41	แดง
32	42	เขียว
33	43	เหลือง
34	44	น้ำเงิน
35	45	ม่วง
36	46	ฟ้า
37	47	ขาว

ตารางที่ 2

```
@echo off
echo This is a NORMAL text
echo This is a [esc][1mBOLD[esc][m text
echo This is a [esc][4mUNDERLINE[esc][m text
echo This is a [esc][5mBLINK[esc][m text
echo This is a [esc][7mREVERSE[esc][m text
echo This is a [esc][8mINVISIBLE[esc][m text
```

รูปที่ 5

เท่านี้ก็พอ

จาก REPLY.COM และคำสั่ง ANSI.SYS 2 คำสั่งที่ผ่านมา คงทำให้คุณเข้าใจ MENU.BAT ได้ดีขึ้น เราจะตกแต่ง MENU.BAT ของเราด้วยการแสดงอักษรเต็มจอภาพเพื่อเป็นพื้นจอ ให้ชื่อไฟล์ว่า BKGR.DOC แล้วเพิ่มคำสั่ง type BKGR.DOC ไว้ในตำแหน่งที่เหมาะสม จากนี้ไป เป็นคำอธิบายเกี่ยวกับ MENU.BAT หากคุณเข้าใจดีแล้วให้ผ่านไป

อธิบาย MENU.BAT

ดู source code ของ MENU.BAT ควบคู่ไปด้วยจะทำให้คุณเข้าใจได้ดีขึ้น

บรรทัดแรก กำหนดให้แบตช์ไฟล์ไม่แสดงคำสั่งที่จะมีต่อไปออกทางหน้าจอ นอกจากนี้ตัวคำสั่ง echo off เองก็มีเครื่องหมาย @ นำหน้าทำให้ไม่แสดงผลทางหน้าจอด้วย

บรรทัดที่ 2-7 กำหนดตัวแปรเอนไวรอนเมนต์ที่จำเป็น บรรทัดที่ว่างไว้เพื่อให้แบตช์ไฟล์ของเราดูเรียบร้อยและสะดวกเวลาตรวจสอบหรือแก้ไขความถูกต้อง

บรรทัดที่ 8-23 เป็นชุดคำสั่ง (โปรแกรมย่อย) ที่แสดงเมนูบนหน้าจอ

- บรรทัดที่ 8 เป็นชื่อเลเบล
- คำสั่ง echo [esc][2J เป็นอีกคำสั่งหนึ่งของ ANSI.SYS ที่ใช้ในการล้างจอภาพ ทำหน้าที่เหมือนคำสั่ง cls แต่ที่ไม่ใช้คำสั่ง cls เนื่องจากต้องการเน้นการใช้งาน ANSI.SYS มากกว่า

การใช้คำสั่งธรรมดาทั่วไป นอกจากคำสั่งล้างจอ ยังมีคำสั่ง {esc}[K ที่ลบข้อความจากตำแหน่งเคอร์เซอร์เรื่อยไปจนกระทั่งสุดจอภาพ (erase to end of line)

- ตัวแปรที่อยู่ในเครื่องหมายเปอร์เซ็นต์ (%) สองตัวเป็นการดึงเอาค่าในตัวแปรนั้น ๆ ออกมาใช้ เช่น echo %msg1% หมายถึงแสดงข้อความ dBASE III Plus... ฯลฯ

บรรทัดที่ 24-26 เป็นชุดคำสั่งที่แสดงข้อความ reverse video ณ บรรทัดตามค่าในตัวแปร row ข้อความดังกล่าวอยู่ในตัวแปร msg เรียบร้อยแล้ว

บรรทัดที่ 27-36 เป็นชุดคำสั่งรับคีย์จากผู้ใช้แล้วตรวจสอบว่าเป็นคีย์ <Esc>, <Enter>, <Arrow-Up> หรือ <Arrow-Dn> หรือไม่

- บรรทัดที่ 28 ย้ายเคอร์เซอร์ไปที่ตำแหน่ง 1;1 เพื่อไม่ให้การแสดงผลเลอะเทอะ

- บรรทัดที่ 29 เรียกใช้โปรแกรม REPLY.COM เพื่อรับคีย์

- บรรทัดที่ 34 ส่งเสียงบี๊บ (โดยการพิมพ์อักขระที่ 7-bell) ในกรณีที่ผู้ใช้กดคีย์อื่น ๆ ที่ไม่ใช่คีย์ที่ต้องการทั้ง 4 ดังกล่าว

- บรรทัดที่ 35 ย้อนกลับไปทำงานที่ต้นวงวน (loop) และจะไม่ออกจากการทำงานจนกว่าผู้ใช้จะกด <Esc>

บรรทัดที่ 37-64 เป็นชุดคำสั่งตรวจสอบค่าในตัวแปร row และ msg แล้วกำหนดให้ถูกต้อง ในกรณีที่ row เป็นบรรทัดแรกหรือบรรทัดสุดท้าย ให้กระโจนไปที่เลเบล Wrap... ทั้งนี้เพราะการทำงานอาจผิดพลาดได้ สมมติว่าค่าของ row เป็น 14 (ค่าสูงสุด) เมื่อผู้ใช้กดคีย์ลูกศรลง (ไปทำงานที่ เลเบล ArrowDn) ถ้าเรากำหนดค่า row ให้เริ่มต้นใหม่คือมีค่าเป็น 12 (ค่าต่ำสุด) ทันที จะทำให้ row ถูกกำหนดเป็น 13 อีกครั้งในบรรทัดที่ 43 ซึ่งไม่ถูกต้อง ด้วยเหตุนี้จึงจำเป็นต้องมีเลเบล WrapUp ขึ้นมาเพื่อให้แบตช์ไฟล์กระโดดไปที่เลเบลดังกล่าว กำหนดค่าให้ตัวแปร row และ msg ใหม่แล้วทำงานต่อไปอย่างถูกต้อง เลเบล ArrowDn จับคู่กับเลเบล WrapUp และเลเบล ArrowUp จับคู่กับเลเบล WrapDn

บรรทัดที่ 65-70 เป็นชุดคำสั่งตรวจสอบค่า row แล้วเรียกใช้แบตช์ไฟล์อื่นเพื่อทำงานต่อไป ในที่นี้คุณสามารถกำหนดแบตช์ไฟล์ของคุณให้เรียกไปที่โปรแกรมที่ต้องการได้ ที่ผมตรวจสอบตัวแปร row เนื่องจากผมไม่ต้องการให้มีตัวแปรมาก เพราะจะทำให้สับสน แต่ในกรณีที่เมนูมีตัวเลือกเยอะ ควรกำหนดตัวแปรเก็บตัวเลือกขึ้นมาต่างหากเพื่อง่ายต่อการตรวจสอบ

บรรทัดที่ 71-78 จะยกเลิกตัวแปรเอนไวรอนเมนต์ทั้งหมด ล้างหน้าจอแล้วแสดงข้อความ * End See you again! ก่อนจบการทำงาน

การเขียนแบตช์ไฟล์เช่นนี้อาจดูเยิ่นเย้อ แต่ทำให้ง่ายต่อการทำความเข้าใจและแก้ไขในภายหลัง ชื่อเลเบลควรเน้นให้ดูง่ายเพื่อแสดงจุดที่แบตช์ไฟล์จะกระโดดไปได้ชัดเจน ควรมีการเว้นบรรทัดเพื่อแบ่งชุดคำสั่งออกจากกัน เหล่านี้เป็นเทคนิคของแต่ละคน

จบซะที

เหนื่อยกันมามาก คุณ ๆ คงตัดพ้อผมว่า ทำไมต้องเรียนรู้อะไรยาก ๆ ด้วย ละ ไม่เห็นจะมีประโยชน์เท่าไรเลย สู้เขียนโปรแกรมก็ไม่ได้

คุณเคยคิดไหมว่าถ้าคุณถูกจำกัดความสามารถ คุณจะหาทางแก้ไขได้อย่างไร จริงอยู่การเขียนภาษาโปรแกรมจะได้โปรแกรมที่มีประสิทธิภาพที่ดีกว่า แต่ถ้าคุณเคยใช้โปรแกรมภาษาใดสักภาษาหนึ่งล่ะก็ คุณเคยใช้งานมันจนปราดเปรื่องเหมือนมือเหมือนเท้าของคุณเอง จนกระทั่ง...คุณไปชนกับขีดจำกัดของภาษา คุณจะทำอย่างไร? คำตอบหนึ่งก็คือ "ศึกษาภาษาใหม่ที่มีขีดความสามารถสูงกว่าสิ เผื่อไว้เยอะ ๆ เลยก็ได้ถ้าอนาคตอาจมีปัญหาอีก" คำตอบเช่นนี้ไม่ใช่วิธีที่ดีเลยครับ คุณต้องเสียเวลา เสียเงิน และอื่น ๆ ในการเรียนรู้สิ่งใหม่อีก คำตอบของผมก็คือ หาโปรแกรมมาช่วย ถึงแม้จะได้ประสิทธิภาพต่ำกว่าภาษารุ่นใหม่ แต่สิ่งสำคัญคือประหยัดทั้งเงิน ทั้งเวลาครับ ซึ่งเวลาช่วงนี้เราสามารถใช้งานได้ประโยชน์อีกมาก อย่างน้อยก็ถ่วงเวลาจนกว่าคุณจะพร้อมทั้งเงินและเวลา สำหรับการทุ่มเทให้กับภาษาใหม่ของคุณ

จุดประสงค์จริง ๆ ของบทความนี้ผมต้องการให้คุณทราบว่า "ภาษาโปรแกรมไม่ใช่สิ่งสำคัญ มันอาจจะทำให้เราเข้าใจการทำงานของโปรแกรมได้ง่ายขึ้น แต่สิ่งสำคัญกว่าคือ ตรรกะ (logic) ในการแก้ไขปัญหาของคุณเอง ถ้าคุณมีสิ่งนี้ล่ะก็ คุณไม่จำเป็นต้องเขียนโปรแกรมเองก็ได้" คุณว่าจริงไหมครับ หมั่นฝึกฝนตรรกะเอาไว้นะครับ